Processes and Operating Systems

- Operating Systems.

This material has been adapted from chapter 6 slides provided with the textbook by Wolf “Computers as Components: principles of embedded system design” Morgan Kaufmann © .

Operating Systems

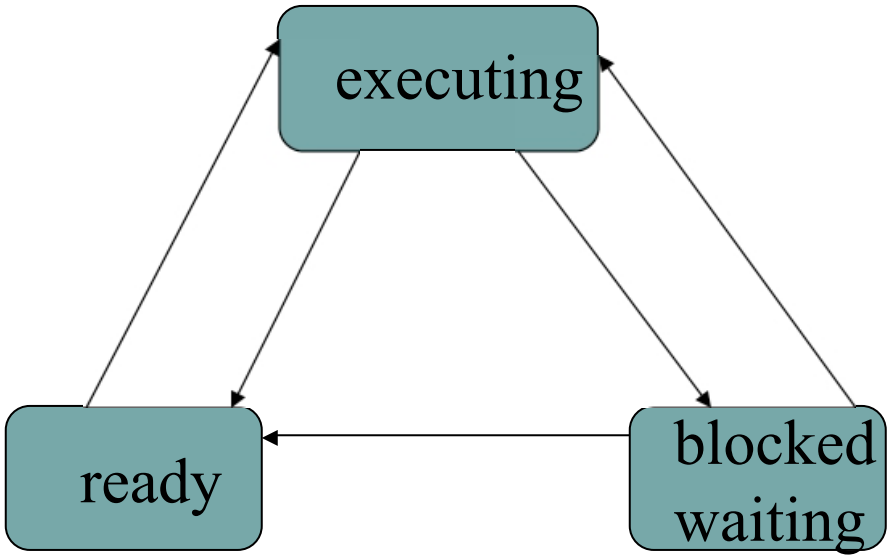
- The operating system controls resources:
  - who gets the CPU;
  - when I/O takes place;
  - how much memory is allocated.
- The most important resource is the CPU itself.
  - CPU access is controlled by the scheduler.

Process state

- After creation a process/task can be in

one of three states:

- executing on the CPU;
- ready to run;
- blocked waiting for data.



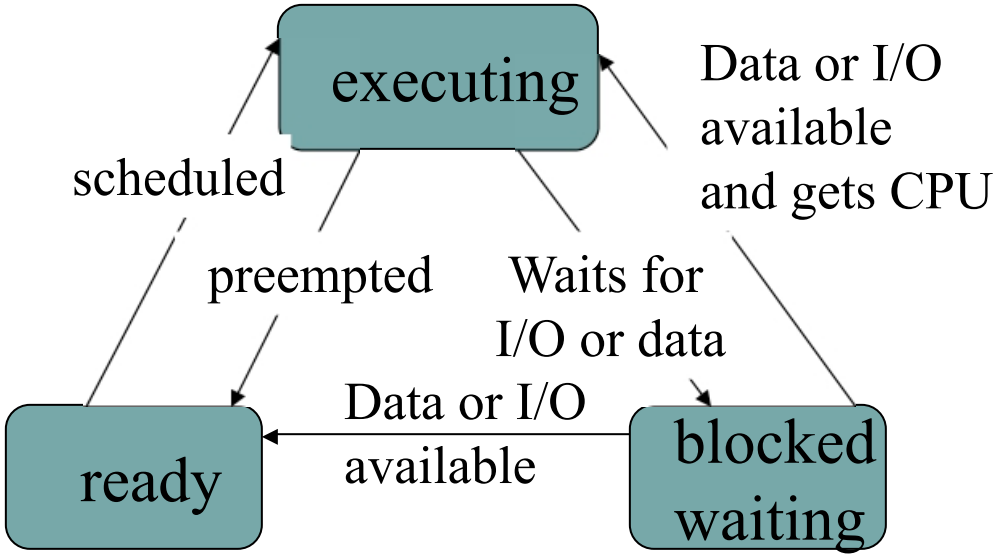
For completeness under a general purpose OS:
There is the New/Entry state
and possibly a Close/Exit state

Process state

- After creation a process/task can be in

one of three states:

- executing on the CPU;
- ready to run;
- blocked waiting for data.



Operating System structure

- OS needs to keep track of:
  - process priorities;
  - scheduling state;
  - process activation record.
- Processes may be created:
  - statically before system starts;
  - dynamically during execution.

Embedded vs. general - purpose scheduling

- Workstations try to avoid starving processes of CPU access.
  - Fairness = access to CPU.
- Embedded systems must meet deadlines.
  - Low -priority processes may not run for a long time.

Time quanta

- Quantum: unit of time for scheduling.

The "system tick": The system tick is the time unit that OS timers and delays are based on. The system tick is a *scheduling event* - i.e. it causes the scheduler to run and may cause a context switch - for example if a timer has expired or a task delay completed.

System Tick:

The system tick interrupt is not the only scheduling event, other mechanisms and events may cause scheduling asynchronously to the system tick.

An RTOS system tick period will typically be in the order of 1ms to 100ms, but may be longer or shorter. The overhead of running the scheduler is increasingly significant the shorter the period, so there is a trade off between timer resolution and CPU overhead.

In many cases real-time response does not rely on timer resolution because events generate interrupts that cause the scheduler to run asynchronously to the clock.

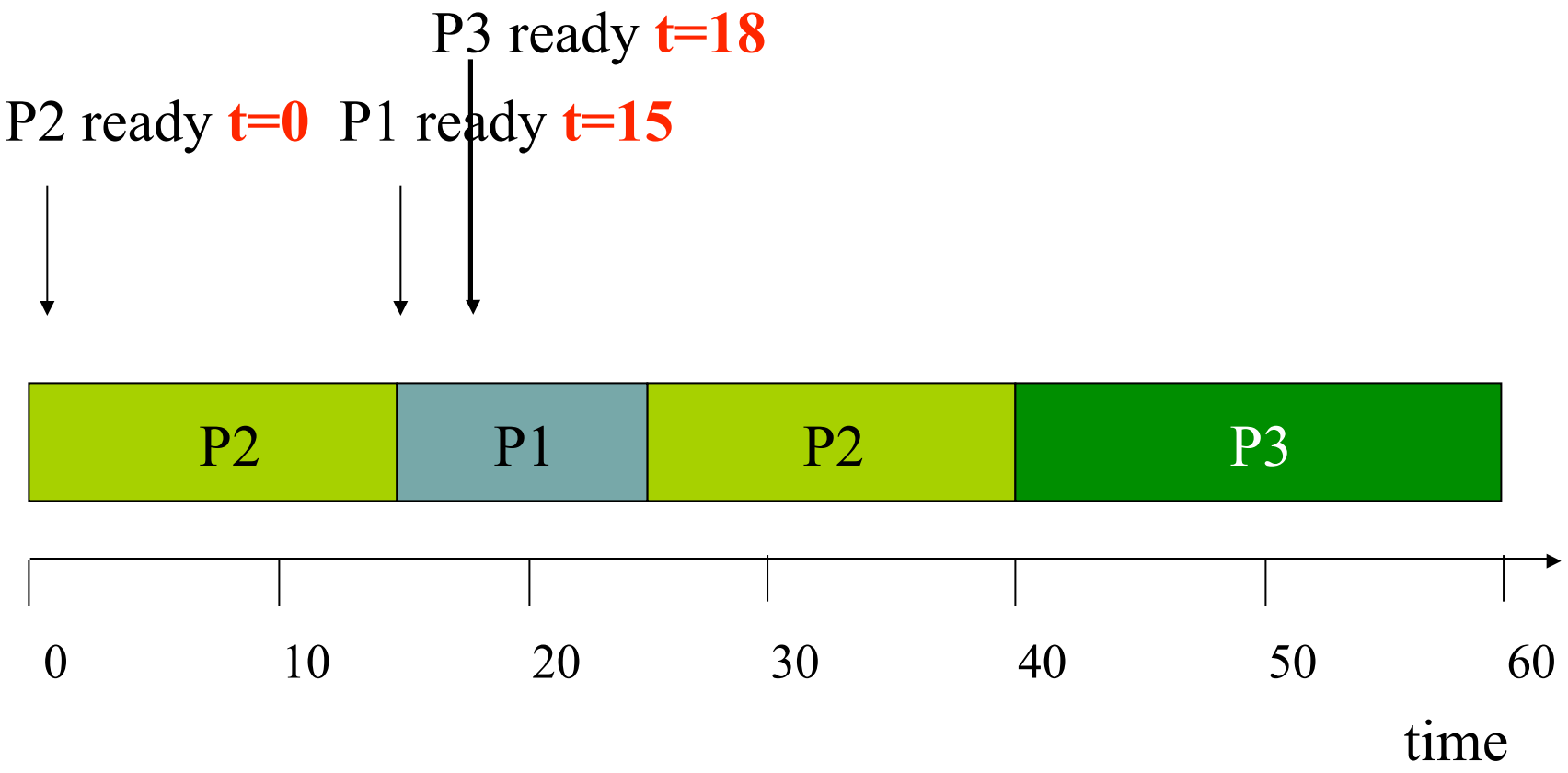
Priority -driven scheduling

- Each process has a priority.
- CPU goes to highest-priority process that is ready.
- Priorities determine scheduling policy:
  - fixed priority;
  - time -varying priorities.

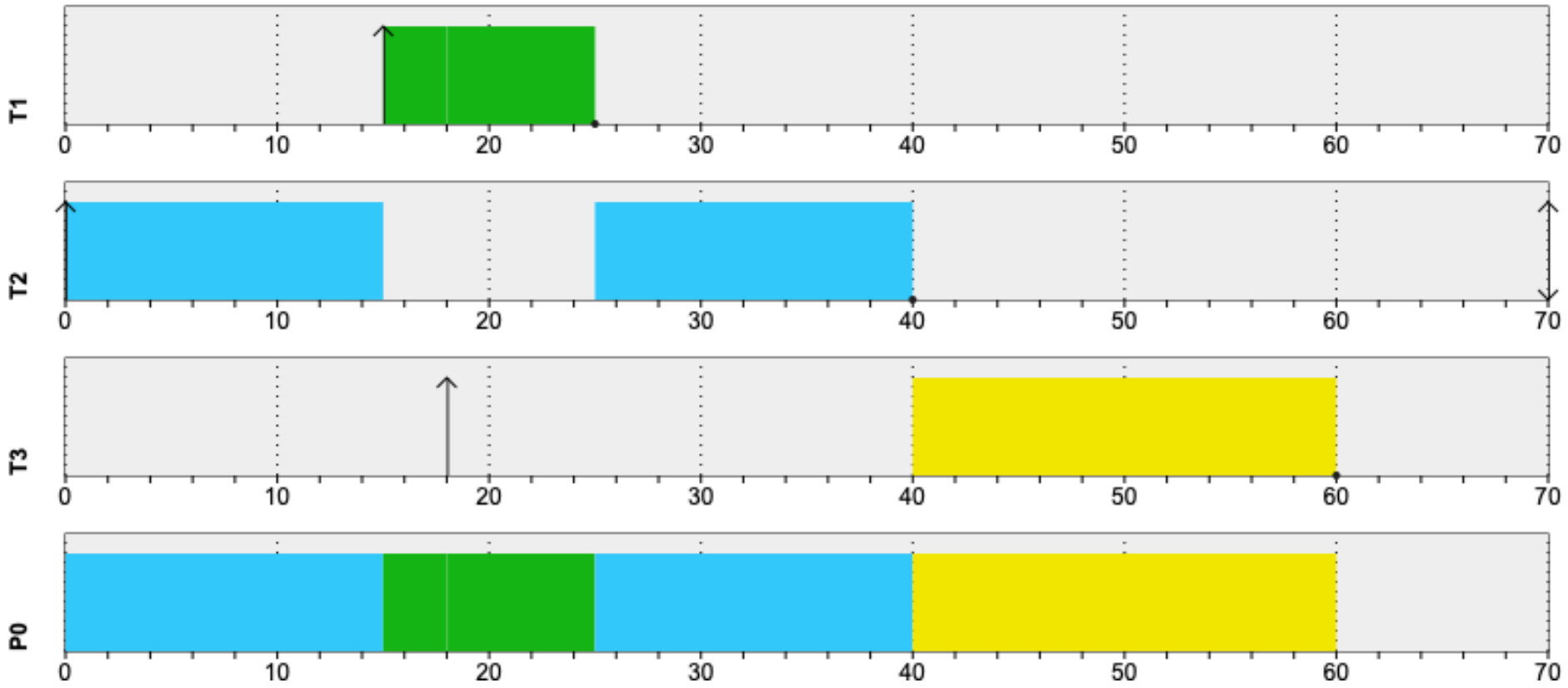
Priority-driven scheduling example

- Rules:
- each process has a fixed priority (1 highest);
- highest -priority ready process gets CPU;
- process continues until done or wait state.
- Processes
- P1: priority 1, execution time 10, ready 15
- P2: priority 2, execution time 30, ready 0
- P3: priority 3, execution time 20, ready 18

Priority-driven scheduling example



Using SimSo for the same set



The scheduling problem

- Can we meet all deadlines?
- Must be able to meet deadlines in all cases.
- How much CPU horsepower do we need to meet our deadlines?

Process initiation disciplines

- Periodic process : executes on (almost) every period.
- Aperiodic process : executes on demand.
- Analyzing aperiodic process sets is harder --- must consider worst-case combinations of process activations.

Timing requirements on processes

- Period : interval between process activations.
- Initiation rate : reciprocal of period.
- Initiation time : time at which process becomes ready.
- Deadline: time at which process must finish.

Timing violations

- What happens if a process doesn't finish by its deadline?
- Hard deadline : system fails if missed.
- Soft deadline : user may notice, but system doesn't necessarily fail.

Example: Space Shuttle software error

- Space Shuttle's first launch was delayed by a software timing error:
  - Primary Avionics Software System PASS and Backup Flight control System BFS.
  - BFS failed to synchronize with PASS.
  - Change to one routine added delay that threw off start time calculation.
  - 1 in 67 chance of timing problem.